Thomas Bennett - trb090020

4395.001 - UTD Spring 2023

Portfolio Assignment 3: WordNet

WordNet is a database of English words grouped by "sets of cognitive synonyms (synsets)." A synset may contain nouns, adjectives, verbs, and adverbs which have some interrelated meaning. WordNet allows programmers to discover the non-ambiguous meanings of otherwise ambiguous English words. For example, 'thread' could be the thin, twisted fiber used for sewing a button onto a garment, or it could be a computational thread, or it could be a verb, the act of threading.
Source: WordNet

In [1]:
```
from nltk.corpus import wordnet as wn

#get all synsets of 'book'

wn.synsets('book')
```

Out[1]:
```
[Synset('book.n.01'),
 Synset('book.n.02'),
 Synset('record.n.05'),
 Synset('script.n.01'),
 Synset('ledger.n.01'),
 Synset('book.n.06'),
 Synset('book.n.07'),
 Synset('koran.n.01'),
 Synset('bible.n.01'),
 Synset('book.n.10'),
 Synset('book.n.11'),
 Synset('book.v.01'),
 Synset('reserve.v.04'),
 Synset('book.v.03'),
 Synset('book.v.04')]
```

In [2]:
```
# Examine the synset: ('book.n.01')

# definition
wn.synset('book.n.01').definition()
```

Out[2]:
```
'a written work or composition that has been published (printed on pages bound togeth
er)'
```

In [3]:
```
# examples
wn.synset('book.n.01').examples()
```

Out[3]:
```
['I am reading a good book on economics']
```

In [4]:
```
# Lemmas
wn.synset('book.n.01').lemmas()
```

Out[4]:
```
[Lemma('book.n.01.book')]
```

In [5]:
```
# Traverse up the WordNet hierarchy
book = wn.synset('book.n.01')
hyper = lambda s: s.hypernyms()
list(book.closure(hyper))
```

Out[5]:
```
[Synset('publication.n.01'),
 Synset('work.n.02'),
 Synset('product.n.02'),
 Synset('creation.n.02'),
 Synset('artifact.n.01'),
 Synset('whole.n.02'),
 Synset('object.n.01'),
 Synset('physical_entity.n.01'),
 Synset('entity.n.01')]
```

WordNet provides a tree structure for nouns. All nouns are descendants of 'entity' whether they are animal, vegetable, or mineral.

In [6]:
```
# The attributes of the book object
dir(book)
```

Out[6]:
```
['__class__',
 '__delattr__',
 '__dict__',
 '__dir__',
 '__doc__',
 '__eq__',
 '__format__',
 '__ge__',
 '__getattribute__',
 '__gt__',
 '__hash__',
 '__init__',
 '__init_subclass__',
 '__le__',
 '__lt__',
 '__module__',
 '__ne__',
 '__new__',
 '__reduce__',
 '__reduce_ex__',
 '__repr__',
 '__setattr__',
 '__sizeof__',
 '__slots__',
 '__str__',
 '__subclasshook__',
 '__weakref__',
 '_all_hypernyms',
 '_definition',
 '_doc',
 '_examples',
 '_frame_ids',
 '_hypernyms',
 '_instance_hypernyms',
 '_iter_hypernym_lists',
 '_lemma_names',
 '_lemma_pointers',
 '_lemmas',
 '_lexname',
 '_max_depth',
 '_min_depth',
 '_name',
 '_needs_root',
 '_offset',
 '_pointers',
 '_pos',
 '_related',
 '_shortest_hypernym_paths',
 '_wordnet_corpus_reader',
 'acyclic_tree',
 'also_sees',
 'attributes',
 'causes',
 'closure',
 'common_hypernyms',
 'definition',
 'entailments',
 'examples',
 'frame_ids',
 'hypernym_distances',
```

```
 'hypernym_paths',
 'hypernyms',
 'hyponyms',
 'in_region_domains',
 'in_topic_domains',
 'in_usage_domains',
 'instance_hypernyms',
 'instance_hyponyms',
 'jcn_similarity',
 'lch_similarity',
 'lemma_names',
 'lemmas',
 'lexname',
 'lin_similarity',
 'lowest_common_hypernyms',
 'max_depth',
 'member_holonyms',
 'member_meronyms',
 'min_depth',
 'mst',
 'name',
 'offset',
 'part_holonyms',
 'part_meronyms',
 'path_similarity',
 'pos',
 'region_domains',
 'res_similarity',
 'root_hypernyms',
 'shortest_path_distance',
 'similar_tos',
 'substance_holonyms',
 'substance_meronyms',
 'topic_domains',
 'tree',
 'usage_domains',
 'verb_groups',
 'wup_similarity']
```

In [7]:

```
nounlist = []

nounlist.append(book.hypernyms())
nounlist.append(book.hyponyms())
nounlist.append(book.part_meronyms())
nounlist.append(book.part_holonyms())

# The book object does not have an antonyms attribute

nounlist
```

Out[7]:
```
[[Synset('publication.n.01')],
 [Synset('appointment_book.n.01'),
  Synset('authority.n.07'),
  Synset('bestiary.n.01'),
  Synset('booklet.n.01'),
  Synset('catalog.n.01'),
  Synset('catechism.n.02'),
  Synset('copybook.n.01'),
  Synset('curiosa.n.01'),
  Synset('formulary.n.01'),
  Synset('phrase_book.n.01'),
  Synset('playbook.n.02'),
  Synset('pop-up_book.n.01'),
  Synset('prayer_book.n.01'),
  Synset('reference_book.n.01'),
  Synset('review_copy.n.01'),
  Synset('songbook.n.01'),
  Synset('storybook.n.01'),
  Synset('textbook.n.01'),
  Synset('tome.n.01'),
  Synset('trade_book.n.01'),
  Synset('workbook.n.01'),
  Synset('yearbook.n.01')],
 [Synset('running_head.n.01'), Synset('signature.n.05')],
 []]
```

In [8]:
```
wn.synsets('execute')
```

Out[8]:
```
[Synset('execute.v.01'),
 Synset('execute.v.02'),
 Synset('carry_through.v.01'),
 Synset('execute.v.04'),
 Synset('run.v.19'),
 Synset('perform.v.01'),
 Synset('execute.v.07')]
```

In [9]:
```
execute = wn.synset('execute.v.01')
execute_def = execute.definition()
execute_example = execute.examples()
execute_lemmas = execute.lemmas()
print(execute_def, execute_example, execute_lemmas, sep='\n')
```

```
kill as a means of socially sanctioned punishment
['In some states, criminals are executed']
[Lemma('execute.v.01.execute'), Lemma('execute.v.01.put_to_death')]
```

In [10]:
```
# Traverse the WordNet hierarchy
hyper = lambda s: s.hypernyms()
list(execute.closure(hyper))
```

Out[10]:
```
[Synset('kill.v.01'), Synset('punish.v.01')]
```

Unlike nouns, verbs do not have a unifying, top-level term like 'entity.' The highest level hypernym for a verb is the most ambiguous form of the verb.

In [11]:
```
word = 'execute'
print(wn.morphy(word, wn.ADJ))
print(wn.morphy(word, wn.NOUN))
print(wn.morphy(word, wn.VERB))
```

```
None
None
execute
```

In [12]:
```
# Two words that may be similar
word1 = 'bottle'
word2 = 'jar'
print(wn.synsets('bottle'), wn.synsets('jar'), sep='\n')
```

```
[Synset('bottle.n.01'), Synset('bottle.n.02'), Synset('bottle.n.03'), Synset('bottle.
v.01'), Synset('bottle.v.02')]
[Synset('jar.n.01'), Synset('jar.n.02'), Synset('jolt.n.01'), Synset('clash.v.02'), S
ynset('jolt.v.01'), Synset('jar.v.03'), Synset('jar.v.04'), Synset('jar.v.05')]
```

In [13]:
```
bottle = wn.synset('bottle.n.01')
jar = wn.synset('jar.n.01')
# WordNet similarity
bottle.path_similarity(jar)
```

Out[13]:
```
0.3333333333333333
```

In [14]:
```
# Wu-Palmer
wn.wup_similarity(bottle, jar)
```

Out[14]:
```
0.8888888888888888
```

In [15]:
```
# Lesk
from nltk.wsd import lesk

sentence = ['The', 'bottle', 'was', 'filled', 'with', 'water', '.']
print(lesk(sentence, 'bottle'))
```

```
Synset('bottle.n.03')
```

In [16]:
```
print(wn.synset('bottle.n.03').definition())
```

```
a vessel fitted with a flexible teat and filled with milk or formula; used as a subst
itute for breast feeding infants and very young children
```

The Wu-Palmer algorithm was able to associate the words 'bottle' and 'jar' with more confidence than the WordNet similarity algorithm. The Lesk algorithm curiously picked the third definition of 'bottle' which refers specifically to a baby's bottle. It could be due to the word 'filled' which is present in the sentence as well as the definition

In [17]:
```
sentence = ['The', 'bottle', 'is', 'full', 'of', 'water', '.']
print(lesk(sentence, 'bottle'))
```

```
Synset('bottle.n.02')
```

In [18]:
```
print(wn.synset('bottle.n.02').definition())
```

```
the quantity contained in a bottle
```

By changing 'was filled with' to 'is full of' caused the Lesk algorithm to choose the second definition of 'bottle' as printed above.

SentiWordNet attempts to examine a word's emotional characteristics. The chosen synset is assigned positivity, negativity, and objectivity scores. SentiWordNet could potentially be used to

help content moderators identify problematic content in a social media website. For instance, a post with a high negatvivity score may need to be removed by a content moderator.

In [19]:
```
for word in wn.synsets('depression'):
    print(word, word.definition(), sep=': ')
```

```
Synset('depression.n.01'): a mental state characterized by a pessimistic sense of ina
dequacy and a despondent lack of activity
Synset('depression.n.02'): a long-term economic state characterized by unemployment a
nd low prices and low levels of trade and investment
Synset('natural_depression.n.01'): a sunken or depressed geological formation
Synset('depression.n.04'): sad feelings of gloom and inadequacy
Synset('depression.n.05'): a period during the 1930s when there was a worldwide econo
mic depression and mass unemployment
Synset('low.n.01'): an air mass of lower pressure; often brings precipitation
Synset('depressive_disorder.n.01'): a state of depression and anhedonia so severe as
to require clinical intervention
Synset('depression.n.08'): a concavity in a surface produced by pressing
Synset('depression.n.09'): angular distance below the horizon (especially of a celest
ial object)
Synset('depression.n.10'): pushing down
```

In [20]:
```
from nltk.corpus import sentiwordnet as swn
depress = swn.senti_synset('depression.n.01')
print(depress)
print('Objectivity: ', depress.obj_score(), sep='')
```

```
<depression.n.01: PosScore=0.0 NegScore=0.375>
Objectivity: 0.625
```

Depression is a widely-discussed issue in the present day. The sentiment analysis is accurate in that the positive score for depression is 0. Knowing the scores of a sentiment analysis could help identify social media content which needs moderating, and could potentially identify individuals requiring therapy.

A collocation is a grouping of two or more words which have a meaning as a group or sequence that they otherwise lack as individuals. Some examples: 'greeting card', 'cookie jar', and 'hard drive.'

In [21]:
```
from nltk.book import text4
text4
```

```
*** Introductory Examples for the NLTK Book ***
Loading text1, ..., text9 and sent1, ..., sent9
Type the name of the text or sentence to view it.
Type: 'texts()' or 'sents()' to list the materials.
text1: Moby Dick by Herman Melville 1851
text2: Sense and Sensibility by Jane Austen 1811
text3: The Book of Genesis
text4: Inaugural Address Corpus
text5: Chat Corpus
text6: Monty Python and the Holy Grail
text7: Wall Street Journal
text8: Personals Corpus
text9: The Man Who Was Thursday by G . K . Chesterton 1908
```

Out[21]:
```
<Text: Inaugural Address Corpus>
```

In [22]:
```
text4.collocations()
```

```
United States; fellow citizens; years ago; four years; Federal
Government; General Government; American people; Vice President; God
bless; Chief Justice; one another; fellow Americans; Old World;
Almighty God; Fellow citizens; Chief Magistrate; every citizen; Indian
tribes; public debt; foreign nations
```

In [23]:
```
import math

text = ' '.join(text4.tokens)

vocab = len(set(text4))
ow = text.count('Old World')/vocab
o = text.count('Old')/vocab
w = text.count('World')/vocab
pmi = math.log2(ow / (o * w))
print('pmi: ', pmi, sep='')
```

```
pmi: 8.983886091037398
```

'Old World' has a high collocation score, since it refers to a vague time and place which was often referenced in the earlier days of the United States. There was a political impetus to show separation between the 'New World' (The United States and the Americas) from the 'Old World' (Europe). There must then be many uses of the phrase 'Old World' in the inaugural corpus as a reference to Europe and Europeans.